

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2001-84235

(P2001-84235A)

(43)公開日 平成13年3月30日 (2001.3.30)

(51)Int.Cl.<sup>7</sup>  
G 0 6 F 15/177  
12/00

識別記号  
6 8 2  
5 7 2

F I  
G 0 6 F 15/177  
12/00

テマコード(参考)  
6 8 2 F 5 B 0 4 5  
5 7 2 A 5 B 0 6 0

審査請求 有 請求項の数 7 O.L. (全 6 頁)

(21)出願番号 特願平11-256766

(22)出願日 平成11年9月10日 (1999.9.10)

(71)出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72)発明者 田辺 純

東京都港区芝五丁目7番1号 日本電気株式会社内

(74)代理人 100088959

弁理士 境 廣巳

Fターム(参考) 5B045 EE03 EE19

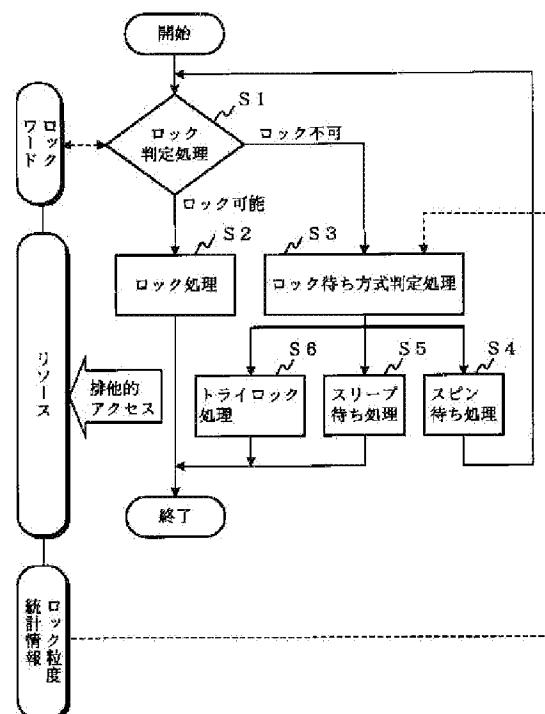
5B060 CD17 KA02 KA04

(54)【発明の名称】 ロック粒度統計情報を用いた排他制御方法及びプログラムを記録した機械読み取り可能な記録媒体

(57)【要約】

【課題】 共有メモリ方式の多重プロセッサ計算機システムにおいて、ロック待ち発生時に発生するオーバヘッド時間を低減させることにより、システム性能を向上させる。

【解決手段】 ロック要求されたリソースRをロックできない場合、リソースRのロック粒度統計情報（ロック時間の平均値等を含む）およびロック要求元がリソースRを使用しない他に行うべき処理を有しているか否かを示すフラグ情報を用いてロック待ち方式をスピン方式、スリープ方式、トライロック方式の何れにするかを決定する。ロック待ち方式をスピン方式にした場合は、更に、ロック粒度統計情報に基づいてスピン待ち時間を決定する。



## 【特許請求の範囲】

【請求項1】 複数のプロセッサがメインメモリを共有する共有メモリ方式の多重プロセッサ計算機システムにおいて、

前記複数のプロセッサによって共有されるリソースがロックされる毎にロック粒度を算出すると共に、該算出したロック粒度に基づいて前記リソースのロック粒度統計情報を更新し、

前記リソースに対するロック待ちが発生したとき、前記リソースのロック粒度統計情報に基づいてロック待ち方式を決定することを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項2】 請求項1記載のロック粒度統計情報を利用した排他制御方法において、

前記ロック粒度統計情報は、ロック粒度の平均値を含むことを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項3】 請求項1記載のロック粒度統計情報を利用した排他制御方法において、

前記ロック粒度統計情報に基づいて決定するロック待ち方式は、スピンドル時間経過後に再度前記リソースに対するロックを試みるスピンドル方式或いはロック要求元をスリープさせるスリープ方式であることを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項4】 請求項3記載のロック粒度統計情報を利用した排他制御方法において、

ロック待ち方式をスピンドル方式と決定した場合は、前記リソースのロック粒度統計情報に基づいて前記スピンドル待ち時間を決定することを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項5】 複数のプロセッサがメインメモリを共有する共有メモリ方式の多重プロセッサ計算機システムにおいて、

前記複数のプロセッサによって共有されるリソースがロックされる毎にロック粒度を算出すると共に、該算出したロック粒度に基づいて前記リソースのロック粒度統計情報を更新し、

前記リソースに対するロック待ちが発生したとき、前記リソースのロック粒度統計情報と、ロック要求元が前記リソースを使用しない処理を行えるか否かを示す情報をに基づいてロック待ち方式を決定し、該決定したロック待ち方式がスピンドル時間経過後に再度前記リソースに対するロックを試みるロック待ち方式である場合は前記リソースのロック粒度統計情報に基づいて前記スピンドル待ち時間を決定することを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項6】 請求項5記載のロック粒度統計情報を利用した排他制御方法において、

前記ロック粒度統計情報に基づいて決定するロック待ち方式は、スピンドル時間経過後に再度前記リソースに

するロックを試みるスピンドル方式、ロック要求元をスリープさせるスリープ方式或いは要求元に他の処理を行わせるトライロック方式であることを特徴とするロック粒度統計情報を利用した排他制御方法。

【請求項7】 コンピュータに、

リソースがロックされる毎にロック粒度を算出すると共に、該算出したロック粒度に基づいて前記リソースのロック粒度統計情報を更新する処理と、

前記リソースに対するロック待ちが発生したとき、前記リソースのロック粒度統計情報に基づいてロック待ち方式を決定する処理とを行わせるためのプログラムを記録した機械読み取り可能な記録媒体。

【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、共有メモリ方式の多重プロセッサ計算機システムにおける排他制御方法に関するもので、特に、排他制御のオーバヘッド時間による計算機システムの性能低下を低減させることができるロック粒度統計情報を利用した排他制御方法に関するものである。

## 【0002】

【従来の技術】 複数のプロセッサがメインメモリを共有する共有メモリ方式の多重プロセッサ計算機システムにおいては、一般に、プロセス（手続き）によるリソース（プロセスが参照や変更などを行う資源や、その資源に対する一連の処理のかたまり）の逐次的（排他的）な使用を保証するため、ロックワードが格納されたメモリ上の領域に対する参照と設定とを1マシンクロックで行うテストアンドセット命令を利用している。ロックワードは、一般的にはメインメモリ上に置かれることが多いが、計算機システムによってはメインメモリではなく特定（専用）のメモリ上に置かれる場合もある。

【0003】 テストアンドセット命令を利用した従来の一般的な排他制御方法は、次のようなものであった。例えば、或るプロセスP aが或るリソースR aの排他的な使用を要求すると、オペレーティングシステムがテストアンドセット命令により、リソースR aに対応するロックワードを参照してリソースR aがロック状態であるか否かを調べると共にリソースR aに対応するロックワードをロック状態を示す値に変更する。

【0004】 そして、ロックワードがロック状態でないことを示している場合は、プロセスP aにロック成功を通知し、リソースR aの排他的な使用を許可する。これに対して、ロックワードがロック状態であることを示している場合には、オペレーティングシステムは、リソースをロックできなかった場合に行う処理（ロック待ち発生時の処理）を行う。ロック待ち発生時に行う処理の方式（ロック待ち方式）には、（1）ロックが取得できるまでスピンドルしながら待ち続けるスピンドル方式、（2）他に行うべき処理がないのでプロセスP aをスリープさせ、他のプロセスのためにプロセッサ（CPU）を開放する

3  
スリープ方式、(3)リソースR'aを使用しない他に行うべき処理があるので、リソースR'aに対するロックをあきらめ、プロセスP'aにリソースR'aを使用しない他の処理を行わせるトライロック方式等がある。

【0005】従来の排他制御方法では、ロック待ち方式は、計算機システム全体あるいはリソース毎に固定的に定められており、オペレーティングシステムは、固定的に定められているロック待ち方式の処理を行う。

【0006】

【発明が解決しようとする課題】一般的に、共有メモリ方式の多重プロセッサ計算機システムでは、最も単純なスピニ方式を採用することが多いが、本質的にどの方式を採用すべきかは、その排他対象となるリソースの性質に依存している。

【0007】ここで、ロック待ちが発生した場合に、計算機システムの性能を落とさないために留意すべき点は、ロックワードに対して必要以上に頻繁にアクセスを行わないようとするということである。すなわち、ロックワードに対して必要以上に頻繁にアクセスを行うと、メモリバンク競合による性能低下を招いてしまうので、そのようなことは避けなければならない。従って、ロック待ちが発生した場合の最も効率的な対処方法は、ロックを取っているプロセスがリソースをアンロックしたタイミングで、タイミングでロックを取得できるような待ち方で一定時間待つ方式である。この待ち時間は、具体的にはそのリソースに対するアクセスパターン、即ちロックとアンロックとのタイミング、さらに言えばロックされていた時間（ロック粒度）に依存していると考えられる。

【0008】ロック粒度が十分に小さい場合は、(1)のスピニ方式が適している。また、ロック粒度がスリープ処理に要する時間（即ちスイッチ処理時間）よりも十分に大きい場合で、なおかつ他に行うべき処理がない場合は、(2)のスリープ方式が適している。逆に、他に行うべき処理がある場合は、(3)のトライロック方式が適しているということになる。つまり、ロック粒度が十分に小さい場合は、CPU時間は無駄に消費されるが、その時間は極僅かなので、制御が簡単で負荷の少ない(1)のスピニ方式が適している。また、ロック粒度がスイッチ処理時間よりも十分に大きい場合は、制御は複雑になるが、CPU時間を有効に使用することができるので、(2)のスリープ方式が適している。

【0009】しかしながら、ロック粒度は、ロック毎に一定の値になるわけではなく、システムの運用状況に応じて動的に変動する。このため、計算機システム全体あるいはリソース毎に固定的にロック待ち方式が定められている従来の排他制御方法では、ロック粒度に応じた最適なロック待ち方式の処理が行われず、その結果、計算機システムの性能を低下させてしまう場合があるという問題がある。例えば、スピニ方式による処理が固定的に

割り当てられているリソースのロック粒度が、システム運用に伴って大きくなった場合には、ロックワードに対するアクセスが必要以上に頻繁に行われ、メモリバンク競合による性能低下を招いてしまう。また、例えば、スリープ方式による処理が固定的に割り当てられているリソースのロック粒度がシステム運用に伴ってスイッチ処理時間よりも小さくなった場合には、スピニ方式に比較して制御が複雑になるだけで、CPU時間を有効利用できなくなってしまう。

10 【0010】そこで、本発明の目的は、ロック粒度の統計情報をを利用してロック待ち方式を動的に変更することで、メモリバンク競合や、無駄なスイッチ処理等によるオーバヘッド時間を減少させることにより、システム全体の性能を向上させることにある。

【0011】尚、特開平4-297949号公報には、競合の発生回数等の統計情報に基づいて、排他単位のサイズを変更する技術が記載されているが、ロック粒度の統計情報に基づいてロック待ち方式を動的に変更するものではない。

20 【0012】  
【課題を解決するための手段】本発明のロック粒度統計情報を利用した排他制御方法は、上記目的を達成するため、複数のプロセッサがメインメモリを共有する共有メモリ方式の多重プロセッサ計算機システムにおいて、前記複数のプロセッサによって共有されるリソースがロックされる毎にロック粒度を算出すると共に、該算出したロック粒度に基づいて前記リソースのロック粒度統計情報を更新し、前記リソースに対するロック待ちが発生したとき、前記リソースのロック粒度統計情報に基づいて

30 ロック待ち方式を決定する。

【0013】ここで、ロック粒度統計情報には、例えば、ロック粒度の平均値が含まれ、また、ロック粒度統計情報に基づいて決定するロック方式には、例えば、スピニ待ち時間経過後に再度リソースに対するロックを試みるスピニ方式あるいはロック要求元をスリープさせるスリープ方式が含まれる。

【0014】また、本発明のロック粒度統計情報を利用した排他制御方法は、ロック方式としてスピニ方式が決定された場合、スピニ待ち時間を最適なものにできるようにするため、ロック待ち方式をスピニ方式と決定した場合は、前記リソースのロック粒度統計情報に基づいて前記ロック待ち時間を決定する。

【0015】  
【発明の実施の形態】次に本発明の実施の形態について図面を参照して詳細に説明する。

【0016】図1は、本発明のロック粒度統計情報を利用した排他制御方法が適用される共有メモリ方式の多重プロセッサ計算機システムの構成例を示すブロック図であり、複数台のプロセッサ（コンピュータ）1-1～1-nと、各プロセッサ1-1～1-nによって共用され

るメインメモリ2と、複数台の入出力処理装置3-1～3-mとから構成される。

【0017】メインメモリ2上には、処理の単位となるプロセス4-1～4-iと、オペレーティングシステム5とが存在する。オペレーティングシステム5は、排他単位となるリソース（メモリ資源やI/O資源）毎の管理テーブル6-1～6-jと、管理テーブル6-1～6-jによって管理される各リソース毎のロックワード7-1～7-j及び待ち行列8-1～8-jとを含んでいる。

【0018】図2、図3はオペレーティングシステム5の処理例を示す流れ図であり、以下各図を参照して本実施例の動作を説明する。

【0019】今、例えば、プロセッサ1-1上で動作しているプロセス4-1において、管理テーブル6-jで管理されているリソースR\_j（図示せず）に対する排他的なアクセスが必要になった場合、プロセス4-1は、リソースR\_jのロックを要求するシステムコールを発行する。尚、このシステムコールには、プロセス4-1において、リソースR\_jを使用しない他に行うべき処理があるか否かを示すフラグ情報が含まれる。

【0020】上記したシステムコールが発行されると、オペレーティングシステム5は、何れかのプロセッサ（例えばプロセッサ1-n）上で、図2の流れ図に示す処理を行う。つまり、オペレーティングシステム5は、プロセッサ1-nによって読み取られ、プロセッサ1-nの動作を制御することで、図2の流れ図に示す処理を行わせる。

【0021】先ず、ステップS1のロック判定処理において、オペレーティングシステム5は、テストアンドセット命令を実行し、リソースR\_jに対するロックワード7-jを参照すると共にロックワード7-jにロック状態であることを示す値を書き込む。そして、参照時のロックワード7-jの値がアンロック状態であることを示している場合、即ちリソースR\_jがロック可能である場合は、ステップS2のロック処理を行い、参照時のロックワード7-jの値がロック状態であることを示している場合、即ちリソースR\_jがロック不可の場合は、ステップS3のロック待ち方式判定処理を行う。

【0022】今、例えば、ステップS1において、ロック可能と判定した場合は、オペレーティングシステム5は、ステップS2のロック処理において、ロック要求元のプロセス4-1に対してロック成功を通知し、更に現在時刻を取得しそれをロック開始時刻としてリソースR\_j対応の管理テーブル6-jに書き込む。

【0023】ロック要求元のプロセス4-1は、ロック成功が通知されると、リソースR\_jを独占的に使用した処理を開始する。

【0024】その後、プロセス4-1が、リソースR\_jのアンロックを要求するシステムコールを発行すると、

オペレーティングシステム5は何れかのプロセッサ（例えばプロセッサ1-n）上で図3の流れ図に示す処理を行う。つまり、オペレーティングシステム5はプロセッサ1-nによって読み取られ、プロセッサ1-nの動作を制御することで、図3の流れ図に示す処理を行わせる。

【0025】先ず、ステップS11のアンロック処理において、オペレーティングシステム5は、アンロック要求されたリソースR\_jに対応するロックワード7-jにアンロック状態を示す値を書き込む。

【0026】次いで、ステップS12のロック粒度算出処理において、リソースR\_j対応の管理テーブル6-jに格納されているロック開始時刻と現在時刻との差分を求ることにより、リソースR\_jがロックされていた時間を（ロック粒度）を算出する。

【0027】その後、ステップS13のロック粒度統計情報算出処理において、管理テーブル6-jに格納されている、リソースR\_jについてのロック粒度統計情報を更新する。ロック粒度統計情報には、ロック回数、ロック粒度の平均値、分散値、最大値、最小値等が含まれており、ステップ13においては、ロック回数を+1したり、ステップS12で算出した今回のロック粒度と管理テーブル6-jに登録されているロック粒度統計情報とに基づいてロック粒度の平均値、分散値、最大値、最小値等を更新する。

【0028】そして、最後にステップS14のウェイクアップ処理において、リソースR\_jのロック待ちをしている、スリープしているプロセスの識別子が登録されている待ち行列8-jを参照し、識別子が登録されている場合は、例えば、先頭に登録されている識別子を取り出し、その識別子のプロセスを起こす。つまり、プロセスをレディ状態にし、CPU時間の割り当て対象にする。

【0029】次に、図2のステップS1において、プロセス4-1によってロック要求されたリソースR\_jをロックできないと判定した場合の動作について説明する。

【0030】ステップS1において、リソースR\_jをロックできないと判定した場合、オペレーティングシステム5は、ステップS3のロック待ち方式判定処理を行い、ロック待ち方式を決定する。

【0031】このステップS3のロック待ち方式判定処理では、例えば、以下のようにしてロック待ち方式を決定する。

【0032】先ず、プロセス4-1が発行したシステムコールに含まれているフラグ情報を参照し、プロセス4-1に、リソースR\_jを使用しない他に行うべき処理があるか否かを調べる。そして、他に行うべき処理がある場合は、ロック待ち方式をトライロック方式とする。これに対して、他に行うべき処理がない場合は、リソースR\_j対応の管理テーブル6-jに格納されているリソースR\_jのロック粒度統計情報に基づいて、ロック待ち方

式をスピン方式にするか、スリープ方式にするかを決定する。例えば、ロック粒度の平均値がスリープ処理に要する時間よりも短い場合はスピン方式、長い場合はスリープ方式、等しい場合はスピン方式、スリープ方式の内の予め定められている一方の方式とする。更に、ロック待ち方式をスピン方式と決定した場合には、管理テーブル6-jに登録されているリソースR jのロック粒度統計情報に基づいて、スピン待ち時間を決定する。例えば、ロック粒度の平均値或いは最小値をスピン待ち間隔とする。

【0033】ステップS 3でロック待ち方式をスピン方式にすることを決定した場合は、オペレーティングシステム5は、ステップS 4のスピン待ち処理を行う。ステップS 4のスピン待ち処理では、ステップS 3で決定されたスピン待ち時間が経過するのを待ち、スピン待ち時間が経過すると再びステップS 1のロック判定処理を行う。

【0034】また、ステップS 3でロック待ち方式をスリープ方式にすることを決定した場合は、ステップS 5のスリープ待ち処理を行う。このステップS 5のスリープ待ち処理では、ロック要求元のプロセス4-1をスリープさせ、プロセス4-1が割り当てられていたプロセッサ1-1を開放する。即ち、プロセス4-1をウエイト状態にすることにより、それが割り当てられていたプロセッサ1-1を開放する。更に、ステップS 3では、プロセス4-1の識別子を、プロセス4-1がロック要求したリソースR jに対応する待ち行列8-jに登録する。

【0035】また、ステップS 3でロック待ち方式をトライロック方式にすることを決定した場合は、ステップS 6のトライロック処理を行う。このステップS 6のトライロック処理では、ロック要求元のプロセス4-1にロック失敗を通知する。この通知を受けたプロセス4-1は、リソースR jを使用しない他の処理を実行する。

【0036】図4は図3のロック粒度統計情報算出処理(ステップS 1.3)および図2のロック待ち方式判定処理(ステップS 3)の処理を詳細に説明するための図である。

【0037】プロセスNがリソースR(図示せず)をアンロックすると、ステップS 1.2のロック粒度算出処理において、ロック粒度Tnを算出し、次のS 1.3のロック粒度統計情報算出処理において、前回までのリソースRのロック粒度統計情報Pn-1と今回新たに算出したロック粒度Tnとから新たなロック粒度統計情報Pnを次式(1)により算出する。

$$Pn = \text{関数} F1(Pn-1, Tn) \quad \dots (1)$$

【0039】ここでロック粒度統計情報Pには、tmin=ロック粒度の最小値、tmax=ロック粒度の最大値、t=ロック粒度の総和、tc=ロック回数、t2=ロック粒度の

2乗の総和、tav=ロック粒度の平均値、tbu=ロック粒度の分散値、その他の情報が含まれている。すなわち、P=(tmin, tmax, t, tc, t2, tav, tbu, その他)である。

【0040】プロセスNがリソースRをロックしているときに、プロセスN+1がリソースRをロックしようとするとき、図2のステップS 1のロック判定処理においてロック不可と判定され、ステップS 3のロック待ち方式判定処理が行われる。このステップS 3のロック待ち方式判定処理では、他に行うべき処理があるかないかを示すフラグ情報flagとロック粒度統計情報Pnとから次式(2)に示すようにしてロック待ち方式Mn+1を決定する。

$$Mn+1 = \text{関数} F2(Pn, flag) \quad \dots (2)$$

【0042】ロック待ち方式をスピン方式と決定した場合には、ロック粒度統計情報Pnから次式(3)に示すようにしてスピン待ち時間Wn+1を決定する。

$$Wn+1 = \text{関数} F3(Pn) \quad \dots (3)$$

【0044】

【発明の効果】以上説明したように、本発明は、ロック粒度統計情報をを利用してロック待ち方式を動的に変更するので、メモリバンク競合や、無駄なスイッチ処理等によるオーバヘッド時間を減少させることができる。この結果、共有メモリ方式の多重プロセッサ計算機システムの性能を向上させることができる。

【0045】更に、本発明は、ロック待ち方式をスピン方式と決定した場合には、ロック粒度統計情報をを利用してスピン待ち時間を決定するようしているので、メモリバンク競合によるオーバヘッド時間を更に減少させることができる。

【図面の簡単な説明】

【図1】本発明方法を適用する共有メモリ方式の多重プロセッサ計算機システムの構成例を示すブロック図である。

【図2】オペレーティングシステム5のロック要求時の処理例を示す流れ図である。

【図3】オペレーティングシステム5のアンロック要求時の処理例を示す流れ図である。

【図4】ロック待ち方式判定処理およびロック粒度統計情報算出処理を説明するための図である。

【符号の説明】

1-1~1-n…プロセッサ

2…メインメモリ

3-1~3-m…入出力処理装置

4-1~4-i…プロセス

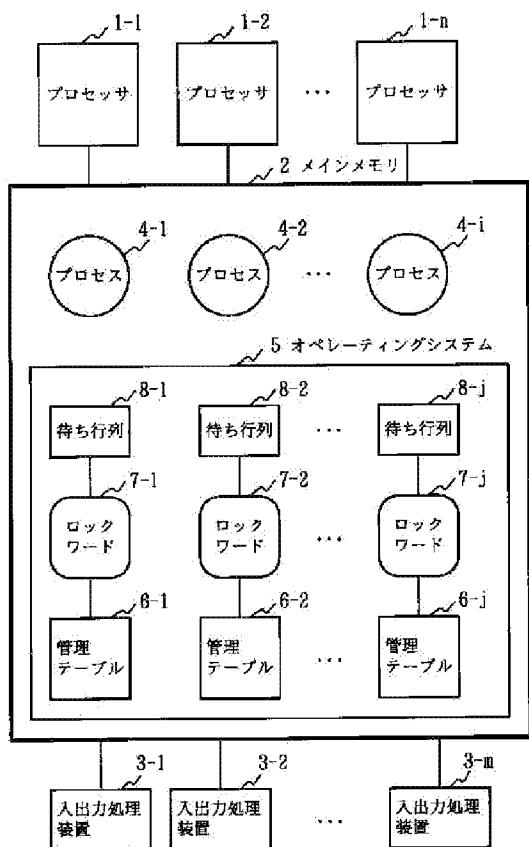
5…オペレーティングシステム

6-1~6-j…管理テーブル

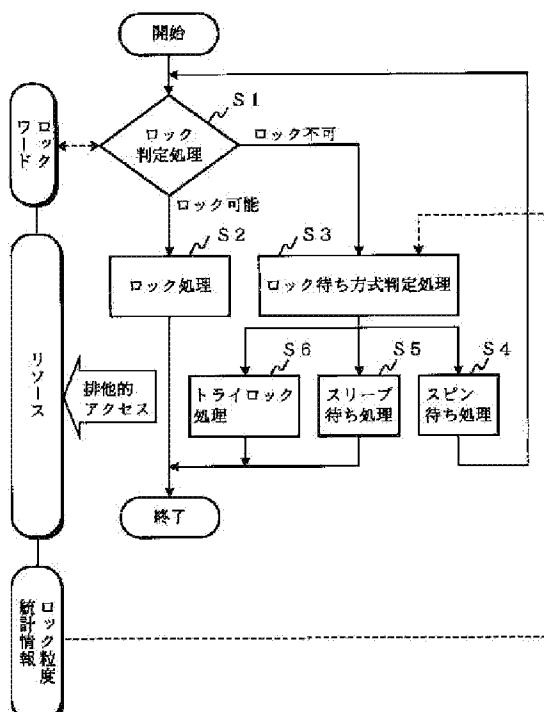
7-1~7-j…ロックワード

8-1~8-j…待ち行列

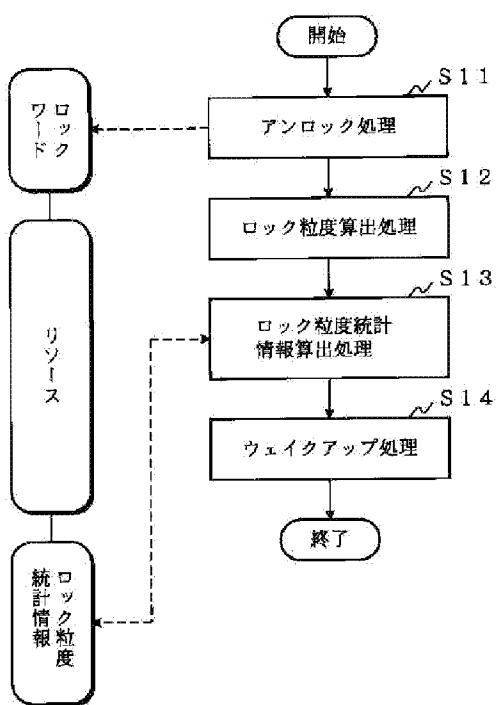
【図1】



【図2】



【図3】



【図4】

